УДК 004.7

В. С. Алексашин, Г. Й. Михальчук

Дніпровський національний університет імені Олеся Гончара

МЕТОД РАНЖУВАННЯ ОБ'ЄКТІВ ДЛЯ ПОШУКУ ЗА КЛЮЧОВИМИ СЛОВАМИ У ДЕЦЕНТРАЛІЗОВАНІЙ МЕРЕЖІ

Розроблено метод ранжування об'єктів для пошуку за ключовими словами у децентралізованій мережі. Представлений метод підтримує 2 види пошуку: пін-пошук та пошук суперсету, які можуть бути використанні для побудови пошукової інфраструктури різного виду застосунків, серед яких: система для пошуку документів, обміну файлами та широкий спектр мультимедійних застосунків.

Ключові слова: *інформаційні технології, децентралізовані мережі, пошук об'єктів, ранжування, розподілені хеш-таблиці.*

Разработан метод ранжирования объектов для поиска по ключевым словам в децентрализованной сети. Представленный метод поддерживает 2 вида поиска: пин-поиск и поиск суперсета, которые могут быть использованы для построения поисковой инфраструктуры разного вида приложений, среди которых: система для поиска документов, обмен файлами и широкий спектр мультимедийных приложений.

Ключевые слова: *информационные технологии, децентрализованные сети, поиск объектов, ранжирование, распределённые хеш-таблицы.*

Search is a core part of any complete file and resource distribution system. Determining the location of unknown files by description – for example, based on keywords or meta-data resources – requires searching. In modern peer-to-peer systems end users cannot retrieve content unless it knows its unique name. In contrast, Web search services, such as Google, allow users to search for content (documents, images, videos, locations etc.). However, these services must actively and repeatedly index the Internet content by hyper-linking from one resource to another. Today, the exponential growth of Internet content makes it difficult to build and maintain a complete index of documents to support effective search. The breadth and rapid development of the Internet means that even the best search services will always be incomplete and inaccurate. The purpose of this work is to develop an efficient peer-to-peer search architecture and algorithm, that support object ranking and superset search. Proposed system involves hypercubes and spanning binomial trees to provide scalable and efficient multiple keyword mechanism, that supports

features, that are highly desired for convenient and effective search infrastructure, while being fully decentralized and fault-tolerant. Dynamic Hash-Tables are used as an overlay network considering their popularity and a list of valuable characteristics. Proposed index scheme uses a clustered approach to group objects based on their keyword sets. This allows you to more evenly distribute workload across network nodes when performing general keyword searches. The described method scales well on a large scale network. In addition, low-level optimizations, such as route caching and search record information on adjacent nodes, can be used to further improve method scaling and reduce overall node load.

Keywords: *peer-to-peer networks, decentralized networks, object search, search ranking, distributed hash-tables.*

Вступ. Peer-to-peer мережі – це самоорганізовані розподілені системи, де вузли-учасники надають та отримують послуги один від одного, не виділяючи ролей чистих клієнтів чи чистих серверів. Peer-to-peer мережі останнім часом привертають велику увагу, насамперед через велику кількість переваг, які вони пропонують для застосунків, побудованих на них. Ці переваги включають масштабованість, доступність, толерантність, децентралізоване управління та анонімність. Поряд з цими бажаними перевагами постала низка технічних проблем.

Пошук – це основна частина будь-якої повної системи розповсюдження файлів та ресурсів. Визначення локації невідомих файлів за описом – наприклад, на основі ключових слів або метаданих ресурсів – вимагає пошуку. У сучасних peer-to-peer системах кінцевий користувач не може отримати контент, якщо він не знає його унікального імені. На відміну від цього, веб-сервіси пошуку дозволяють користувачам здійснювати пошук за вмістом. Однак ці сервіси повинні активно та неодноразово індексувати інтернет-вміст, слідуючи за гіперпосиланням з одного ресурсу на інший. Поточна пошукова інфраструктура в Інтернеті обмежена її централізованим дизайном та принципом HTTP, що заснований на стягуванні даних.

Поточний рух у напрямку peer-to-peer систем для розповсюдження файлів дає нам можливість значно покращити пошук. Якщо пошук в Інтернеті є централізованим, peer-to-peer пошук може бути повністю розподіленим, і, таким чином, більш масштабованим та надійним. Якщо пошук в Інтернеті залежить від сканерів для відкриття нового або оновленого контенту, система децентралізованого пошуку може скористатися операціями вставки для створення поточного індексу всього мережевого контенту.

Хоча, мабуть, можна створити високоефективну централізовану службу пошуку для peer-to-peer систем, пошукова інфраструктура для розподіленого peer-to-peer сховища також повинна бути розподілена для досягнення описаних вище переваг. Використовуючи ту саму надлишкову, розподілену інфраструктуру, яка підтримує peer-to-peer зберігання та пошук вмісту, можна досягти збільшеної доступності, масштабованості та балансування навантаження.

Аналіз пов'язаних досліджень. Перше покоління однорангових систем складається з Napster [11], Gnutella [12] та Freenet [13]. Napster і Gnutella використовують пошук як основну техніку визначення місця розташування. Napster здійснює пошук централізовано на відомих серверах, які зберігають метадані, місцеположення та ключові слова для кожного документа. Gnutella здійснює трансляцію пошукових запитів на всі вузли та дозволяє кожному вузлу здійснювати пошук у конкретному виконанні. Freenet не забезпечує механізму пошуку, а натомість залежить від відомих імен та відомих каталогів імен.

Існуючі механізми пошуку peer-to-peer даних, такі як Kadelemia [2], Chord [4], CAN [5], PAST [6], фактично забезпечують масштабовану розподілену хеш-таблицю, яка дозволяє відображати окремі ключі до вузлів по всій мережі. Коли вузол бажає опублікувати вміст, він спочатку хешує вміст файлу, щоб виявити вузол, відповідальний за його збереження. Таким чином неможливо здійснити пошук, не знаючи точного ідентифікатора файлу.

Для забезпечення можливості пошуку за допомогою ключових слів запропоновано розширення, яке називається методом перевернутої розподіленої хеш-таблиці [3]. Основна ідея перевернутої розподіленої хеш-таблиці полягає у використанні ключових слів як індексів для пошуку файлів шляхом збереження відображення <ключове_слово-список_значень> на кожному вузлі розподіленої хеш-таблиці замість <ідентифікатор_файлу-значення>.

Система, яка використовує механізм перевернутої хеш-таблиці, є слабкою до проблеми, що називається загальною проблемою ключових слів. Загальна проблема ключових слів виникає через те, що певні ключові слова зазвичай асоціюються з дуже великою кількістю файлів порівняно з іншими ключовими словами. Отже, невеликій кількості сусідніх вузлів, які відповідають за збереження інформації про місцеположення для таких загальних ключових слів, доведеться споживати надмірну кількість сховища та мережевого трафіку, порівняно з іншими вузлами. Для подолання цієї проблеми було запропоновано використання фільтра Блума [3] та Keyword Fusion [8].

Зазначені вище роботи не вирішують проблему ранжування об’єктів, яка досі є недостатньо вивченою. Якщо простір об’єктів є дуже великим, запит, що складається з декількох популярних ключових слів, може отримати набір з дуже великою кількістю об’єктів. Ранжування загалом вимагає певних глобальних знань. Наприклад, при пошуку інформації для вимірювання важливості ключового слова використовується концепція зворотної частоти документа (IDF). Він визначається як логарифм відношення кількості документів у колекції до кількості документів, що містять ключове слово [10]. Тож нечасті слова мають високий показник IDF, а загальні слова, наприклад "файл", мають низький рівень IDF. IDF можна легко обчислити, коли служба індексації централізована, але в децентралізованому середовищі вартість обчислення якісної оцінки є дуже високою.

Постановка задачі. Метою роботи є розробка ефективного методу ранжування об’єктів для пошуку за ключовими словами у децентралізованій мережі, що підтримує пошук суперсету з динамічним значенням порога.

Неформальний опис методу. Основна ідея методу полягає в тому, щоб представити кожен об’єкт у вигляді бітового вектора відповідно до його набору ключових слів. Розглядаючи ці r-бітові вектори як точки в r-мірному гіперкубі, можна отримати багато корисних властивостей.

По-перше, записи в індексі одного ключового слова обробляються набором вузлів. Кількість вузлів цього набору залежить від популярності ключового слова: чим більше популярність ключового слова, тим більше кількість вузлів, що відповідають за ключове слово. Як результат, навантаження вузлів може бути врівноважене, і жоден вузол, ймовірно, не може бути перевантажений, навіть якщо він обробляє дуже популярне ключове слово. Крім того, оскільки низка вузлів відповідає за одне ключове слово, будь-який збій в них не може блокувати запити за участю ключового слова.

По-друге, об’єкт σ, пов’язаний із набором ключових слів K, може бути ефективно та детерміновано розміщений, якщо задано множину K. Це аналогічно пошуку ідентифікатора в мережах DHT. Як обговорювалося раніше, мережі DHT використовують ідентифікатор об’єкта для визначення його вузла обробки. Після цього розміщення об’єкта – це просто маршрутизація повідомлення до вузла, що може бути зроблено дуже ефективно в мережах. У запропонованій схемі індексів використано набір ключових слів, що асоціюється з об’єктом, для визначення унікального вузла для індексації об’єкта. Коли набір

відомий, розміщення об'єкта є настільки ж ефективним, як пошук імен у DHT-мережах.

По-третє, в пошуковій операції, крім об'єктів, набори ключових слів яких точно збігаються із заданим набором ключових слів K, можна також побачити об'єкти, набори ключових слів яких містять K. Усі ці об'єкти можна легко та ефективно отримати в нашій індексній схемі. Більше того, чим більший набір K вказав користувач, тим більше обмеження зробив користувач на своїх цільових об'єктах. Відповідно, запропонована індексна схема вимагатиме контакту з меншою кількістю вузлів. З іншого боку, коли задано невеликий набір K, велика кількість об'єктів може задовольнити запит пошуку. У цьому випадку користувач часто розраховує побачити лише невелику їх підмножину. Запропонована індексна схема також може ефективно підтримувати цей вид операцій.

У вищеописаній пошуковій операції, коли кількість відповідних об'єктів велика і користувач розраховує побачити лише частину з них, запропонована індексна схема полегшує різноманітні способи допомогти користувачам класифікувати об'єкти, що співпадають. Зокрема, об'єкти в такій індексній схемі легко розрізнити за кількістю пов'язаних ними ключових слів. Наприклад, нехай K – це набір ключових слів. Запропонована індексна схема може легко знаходити об'єкти, які асоціюються саме з набором K ключових слів, об'єктами, пов'язаними з K плюс ще одним ключовим словом, K плюс ще двома ключовими словами тощо. Більше того, у межах кожної категорії, наприклад, K плюс ще одне ключове слово, об'єкти можуть відрізнятись додатковим ключовим словом, наприклад, K плюс певним ключовим словом σ_1, K плюс певним ключовим словом σ_2 тощо.

Ця цікава функція дозволяє застосункам верхнього рівня отримувати об'єкти у бажаному порядку. Наприклад, запит може віддавати перевагу спочатку більш конкретним об'єктам. У цьому випадку, коли видається запит на пошук із набором ключових слів K, індексна схема може повертати об'єкти, що містять цей набір ключових слів K, у порядку, надаючи перевагу тим, у кого більше додаткової кількості ключових слів. Для впровадження цього механізму ранжування не потрібні глобальні знання.

Розподілена хеш-таблиця. Розподілена хеш-таблиця (DHT) – це клас децентралізованої розподіленої системи, що утворює накладну мережу, яка за своєю функцією, схожа на хеш-таблицю: пари <ключ-значення> зберігаються в DHT, і будь-який вузол, що бере участь, може

ефективно отримати значення, пов'язане із заданим ключем. Ключі – це унікальні ідентифікатори, які відображають певні значення, які, у свою чергу, можуть бути будь-якими – від адрес та документів до довільних даних [4].

Накладна мережа може бути змодельована спрямованим графом $G=(V,E)$, де V – сукупність вузлів у мережі, а E – набір зв'язків між вузлами. Кожному вузлу u присвоюється бітова унікальна ідентифікація. Коли двозначність неможлива, лише u може бути використана для ідентифікації. Наявність ребра (u,v) в E означає, що вузол u знає прямий спосіб надіслати повідомлення вузлу v.

Набір об'єктів Ω поділяється між мережею. Кожен об'єкт $\sigma \in \Omega$ також має унікальний ідентифікатор і має набір реплік, поширених у мережі. Для кожного вузла u, який зберігає копію σ, використаємо (σ,u) для позначення посилання на репліку. Посилання (σ,u) зазвичай складається з IP/порту u та фізичної адреси σ у вузлі. Щоб отримати доступ до копії σ, слід отримати посилання на σ.

Пов'язаною з накладною мережею G є схема розподіленої локалізації та маршрутизації об'єктів (DOLR) для доступу до об'єктів у мережі. Схема DOLR включає відображення $L:\Omega \to 0,1,\ldots,2^a-1$ та один механізм маршрутизації. Відображення L детерміновано та рівномірно відображає кожен об'єкт $\sigma \in \Omega$ (за його ідентифікатором) у точно один вузол – представлений у вигляді бітового бінарного рядка – для обробки об'єкта. Механізм маршрутизації визначає для кожної пари вузлів u і v з V принаймні один (u,v) шлях у G.

Операції вставки, видалення та читання в схемі DOLR вимагають того, щоб був відомим ідентифікатор цільового об'єкта. У деяких застосуваннях може знадобитися можливість знаходити об'єкт за лише ключовими словами (атрибутами). Для надання послуги пошуку ключових слів у накладній мережі вважаємо, що кожен об'єкт $\sigma \in \Omega$ асоціюється з набором K_σ ключових слів. Для будь-якого об'єкта σ говоримо, що набір ключових слів K може описувати σ, якщо $K \subseteq K_\sigma$.

Для кожного набору K ключових слів визначаємо набір Ω_K об'єктів, де $\Omega_K = \{\sigma \,||\, \sigma \in \Omega, K \subseteq K_\sigma\}$. Тобто Ω_K – це сукупність об'єктів, яку можна описати за допомогою K. Розмір $||\Omega_K||$

називається ключовою частотою K.

Для надання сервісу пошуку за ключовими словами потрібно розробити розподілену схему індексу, в якій об'єкт може бути розташований, вказавши в запиті декілька ключових слів. У сервісі можна виділити дві функції:

1. Пін-пошук: з урахуванням набору ключових слів K служба повинна повернути набір $\{\sigma \,||\, K_\sigma = K\}$ об'єктів, які асоціюються саме з набором ключових слів K.
2. Пошук суперсету: з огляду на набір ключових слів K та деякий поріг t, сервіс повинен повернути набір $\min(t, ||\Omega_K||)$ об'єктів, які можуть бути описані за допомогою K.

На практиці пошук суперсету може бути розглянутий як сукупний, коли результати, повернуті послідовними пошуками з одним і тим самим набором ключових слів, повинні бути різними. Зазвичай це використовується у великих інформаційних системах, таких як Google, у яких користувачі можуть "переглядати" великі набори об'єктів крок за кроком.

Механізм пін-пошуку. Запропонована розподілена індексна схема побудована на логічній структурі – r-мірному просторі гіперкубів – через мережу DHT. Щоб представити цю схему індексів, спочатку вводимо гіперкуб, а потім взаємозв'язок між логічною структурою та базовою мережею. R-розмірний гіперкуб $H_r(V,E)$ має 2^r вузлів. Кожен вузол u в V представлений унікальним r-бітним двійковим рядком. Використаємо $u[i]$ для позначення i-го біта u (рахуючи справа). Для кожної пари вузлів (u,v) у V і кожного цілого числа i в $\{0,1,\ldots,r-1\}$, у E існує (непряме) ребро (u,v), якщо і тільки якщо u відрізняється від v лише в i-му біті. Говоримо, що ребро (u,v) перетинає i-й вимір, а u – сусід v у i-му вимірі, і навпаки.

Для кожного вузла $u \in V$ визначаємо набір $One(u)$ цілих чисел так: $One(u) = \{i \,||\, u[i] = 1, 0 \le i \le r-1\}$; тобто позиції, в яких u біти містять 1. Аналогічно визначаємо $Zero(u) = \{i \,||\, u[i] = 0, 0 \le i \le r-1\}$. Наприклад, якщо $v = 0b010100$, то $One(v) = \{2,4\}$ і $Zero(v) = \{0,1,3,5\}$.

Трансляція може бути здійснена дуже ефективно в гіперкубах за допомогою використання охоплюючих біноміальних дерев [1]. Охоплююче біноміальне дерево $\mathrm{SBT}(u)$ для гіперкуба H_r має глибину r, а вузол v на m-му рівні $\mathrm{SBT}(u)$ (де корінь u знаходиться на рівні 0) має відстань Хеммінга m від u. Відстань Хеммінга між будь-якими двома r-бітовими двійковими рядками u та v є $\mathrm{Hamming}(u,v) = \Sigma_{i=0}^{r-1}(u[i] \oplus v[i])$. Тобто в охоплюючому біноміальному дереві кожен вузол, який знаходиться на глибині i від кореня, має точно i бітів, що відрізняються від кореня за своїм бінарним поданням. Ця важлива властивість пізніше буде використана в нашій схемі пошуку ключових слів. Можемо аналогічно визначити охоплююче біноміальне дерево на $H_r(u)$, маскуючи біти, що виникають у положеннях в $\mathrm{One}(u)$ для кожного вузла в $H_r(u)$. У цій роботі використаємо охоплююче біноміальне дерево з коренем в u на $H_r(u)$. Назвемо це охоплюючим біноміальним деревом, створеним від u, і зазначимо його $\mathrm{SBT}_{H_r}(u)$.

Нехай W – сукупність усіх ключових слів, що розглядаються в системі. Нехай $h: W \to \{0,1,\ldots,r-1\}$ – рівномірна хеш-функція, яка відображає кожне ключове слово у W на ціле число в $\{0,1,\ldots,r-1\}$. Визначимо відображення $F_h: 2^W \to V$ таким чином: $F_h(K) = u$, якщо і тільки якщо $\mathrm{One}(u) = \{i \,||\, h(\sigma) = i, \sigma \in K\}$. Іншими словами, $F_h(K)$ – вузол з бінарним числом, біти якого задаються хеш-функцією h відповідно до ключових слів у K. Говоримо, що u відповідає за K, якщо $F_h(K) = u$. Таким чином, для кожного можливого набору ключових слів у системі існує вузол у гіперкубі, відповідальний за набір. Зауважимо, що вузол може відповідати за більш ніж один набір ключових слів, оскільки $F_h(K)$ може дорівнювати $F_h(K')$ для деяких K і K'. Позначимо набор ключових слів, за які відповідає вузол u, через R_u. Тобто $R_u = \{K \subseteq W \,||\, F_h(K) = u\}$.

Для побудови схеми індексу для кожного об’єкта σ, асоційованого

з набором ключових слів K_σ, дозволяємо вузлу $F_h(K_\sigma)$ у гіперкубі підтримувати запис $\langle K_\sigma, \sigma \rangle$ у своїй індексній таблиці. Говоримо, що σ індексується у вузлі, і використовуємо Ω_u для позначення набору об'єктів, які індексуються у u. Тобто $\Omega_u = \sigma \in \Omega \,||\, K_\sigma \in R_u$.

Враховуючи набір ключових слів K, можливо знайти копію об'єкта, пов'язаного з K, спочатку знаходячи вузол у H_r, відповідальний для K. Вузол визначається за $F_h(K)$. Після того як вузол буде знайдений, за допомогою його таблиці індексів можна отримати ідентифікатор об'єкта σ, який асоціюється з набором ключових слів K. Тож пін-пошук безпосередньо підтримується даною схемою.

Механізм пошуку суперсету. Операція пошуку суперсету, що складається з набору ключових слів K і деякого порогу t, повинна повернути набір об'єктів $\min(t, ||\Omega_K||)$, який може бути описаний за допомогою K. Простір пошуку може бути обмежений у субгіперкубі $H_r(F_h(K))$. Хоча будь-який вузол субгіперкуба може потенційно індексувати деякі об'єкти, які можна описати K, між ними є незначна різниця. Якщо вузол x знаходиться на відстані d від коренів у охоплюючому біноміальному дереві $SBT_{H_r}(F_h(K))$, то кожен об'єкт, індексований на x, який може бути описаний K, асоціюється з набором ключових слів, що містить щонайменше на d ключових слів більше, ніж K. Тому такий об'єкт, ймовірно, буде менш загальним (і, таким чином, більш конкретним для певного предмета), ніж об'єкти з точно встановленим ключовим словом K. Залежно від додатків, можна досліджувати охоплююче біноміальне дерево у ширину зверху-вниз або знизу-вгору. Перший повертає результати пошуку, надаючи перевагу більш загальним об'єктам, а другий – більш конкретним. Далі описано перший підхід, але друга альтернатива може бути зроблена лише з незначною модифікацією.

Нехай $v = F_h(K)$ – вузол, який відповідає за K. Вузол u, який ініціює запит на пошук суперсету, надсилає до v запит на ключове слово $Query(K, t, u, -, -)$, де K – це запитане ключове слово, t – поріг, а u – прямий контакт до вузла u, що збирає результати. Інші два аргументи не використовуються в початковому запиті, але будуть

використані пізніше під час операції. Коли v отримує запит, він використовує свою таблицю індексів, щоб знайти всі записи $\langle K', O\rangle$ з $K' \supseteq K$, а потім надсилає ідентифікатори об'єктів (до t ідентифікаторів) безпосередньо до u. Операція пошуку закінчується, якщо повернуто t ідентифікаторів об'єкта. В іншому випадку v записує залишкову кількість ідентифікаторів об'єктів, які потрібно зібрати, в лічильник c. Вузол v також ініціалізує чергу U пар $(x,d), 0 \le d \le r-1$, де x є прямим контактом до вузла x в $SBT_{H_r}(v)$, а d – індекс для обчислення дітей вузла x у $SBT_{H_r}(v)$. Черга U ініціалізується таким чином:

Для кожного i в $Zero(v)$, (y,i) додається до U, де y – сусід v і-му вимірі. Потім операція пошуку виконується за вказаними нижче кроками:

1. Якщо U не порожня, перша пара (w,d) видаляється з U, а v посилає w повідомлення $Query(K,c,u,d,v)$. Потім вузол v переходить до кроку 3, щоб чекати результату. Якщо U порожня, пошукова операція припиняється.

2. Для кожного вузла w, який отримує повідомлення $Query(K,c,u,d,v)$, w спочатку вивчає свою таблицю індексів, щоб знайти всі записи $\langle K', O\rangle$ з $K' \supseteq K$, а потім надсилає ідентифікатори об'єктів (до c ідентифікаторів) безпосередньо до u. Нехай c_1 – кількість повернених ідентифікаторів об'єкта. Операція пошуку може припинитися, якщо $c_1 \ge c$, і в цьому випадку w посилає v повідомлення Stop, щоб повідомити v про це. В іншому випадку w готує список L, що складається з таких пар: $\{(x,i) \,||\, i < d \wedge i \in Zero(w)$, а x – сусід у і-му вимірі$\}$. Тоді w посилає повідомлення $Cont(c_1, L)$ на v.

3. Вузол v завершує операцію пошуку після отримання повідомлення Stop від w. В іншому випадку він чекає, поки w відправить йому повідомлення $Cont(c_1, L)$. Потім v додає пари з L до U, встановлює c в $c - c_1$ і переходить до кроку 1 для продовження операції. Пошук суперсету може бути накопичувальним, щоб

послідовні пошукові запити переглядали великі відповідні набори кроків. У вищеописаній операції сукупний пошук суперсету може бути легко реалізований, дозволяючи кореневому вузлу $v = F_h(K)$ зберігати чергу U для наступних запитів, поки пошук не завершиться.

Безпека DHT. Через децентралізацію, толерантність до відмов та масштабованість DHT, вони за своєю суттю більш стійкі до атаки, ніж типова централізована система. Система DHT, яка ретельно розроблена для забезпечення візантійської відмовостійкості, може захищати від слабкості в безпеці, відомої як атака Sybil, яка впливає на всі існуючі конструкції DHT [9]. Петро Меймунков, один з оригінальних авторів DHT Kadelemia [2], запропонував спосіб обійти слабкість до нападу Sybil, включивши в систему системи відносини соціальної довіри. Нова система під кодовою назвою Tonika, або відома також своїм доменним іменем 5ttt, заснована на розробці алгоритму, відомого як Electric routing, у співавторстві з математиком Джонатаном Келнером [7]. Однак дослідження ефективних засобів захисту від нападів Sybil, як правило, вважається відкритим питанням, і щорічно на найвищих науково-дослідних конференціях з питань безпеки пропонується широкий спектр потенційних засобів захисту.

Висновки. Розроблено метод ранжування для пошуку об’єктів за ключовими словами у децентралізованій мережі. Представлений метод підтримує 2 види пошуку: пін-пошук та пошук суперсету, які можуть бути використанні для побудови пошукової інфраструктури різного виду застосунків, серед яких система для пошуку документів, обміну файлами та широкий спектр мультимедійних застосунків. На відміну від підходу з використанням розподіленого інвертованого індексу, запропонована схема індексів на основі гіперкубів використовує кластерний підхід до групування об’єктів на основі їх наборів ключових слів. Це дозволяє більш рівномірно розподілити навантаження між вузлами мережі при виконанні пошуку за загальними ключовими словами. Описаний метод добре масштабується на мережі великого масштабу. Крім того, оптимізації низького рівня, такі як кешування маршрутів та інформації про пошукові записи на сусідніх вузлах, можуть бути використані для подальшого покращення масштабування методу та зниження загального навантаження на вузли.

Бібліографічні посилання

1. Johnsson S. L., Ho C. T. Optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. on Computers*. 1989. 38(9). C. 1249–1268.

2. Maymounkov P., Mazières D. Kademlia: A Peer-to-peer information system based on the XOR Metric. *IPTPS 2002: Peer-to-Peer Systems*. 2002. C. 53–65.

3. Reynolds P., Vahdat A. Efficient Peer-to-Peer Keyword Searching. *Middleware*. 2003. P. 21–40.

4. Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. *Proceedings of ACM SIGCOMM'01*. 2001. P. 149–160.

5. Ratnasamy T., Francis P., Handley M., Karp R., Shenker S. A scalable content-addressable network. *Proceedings of ACM SIGCOMM'01*. 2001. P. 161–172.

6. Rowstron A., Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *ACM Symposium on Operating Systems*. 2001. P. 188–201.

7. Kelner J., Maymounkov P. Electric routing and concurrent flow cutting. *Theoretical Computer Science*. 2011. Vol. 412. P. 4123–4135.

8. Liu L., Ryu K. D., Lee K. W. Supporting Efficient Keyword-based File Search in Peer-to-Peer File Sharing Systems. *IEEE Global Telecommunications Conference*. 2004. P. 1259–1265.

9. Young M., Kate A., Goldberg I., Karsten M. Practical Robust Communication in DHTs Tolerating a Byzantine Adversary. *ICDCS '10 Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*. 2010. P. 263–272.

10. Jones K. S. Index term weighting. *Information Storage and Retrieval*. 1973. P. 619–633.

11. Napster. URL: http://www.napster.com/ (дата звернення: 04.11.2019).

12. Gnutella. URL: http://gnutella.wego.com/ (дата звернення: 04.11.2019).

13. Hong T. Freenet: A distributed anonymous information storage and retrieval system. *ICSI Workshop on Design Issues in Anonymity and Unobservability*. 2000. P. 46–66.

Надійшла до редколегії 15.11.2019.